4DS4

Lab 3/Project2

Autonomous Vehicle: Putting it All Together

Submitted by:

Group 18

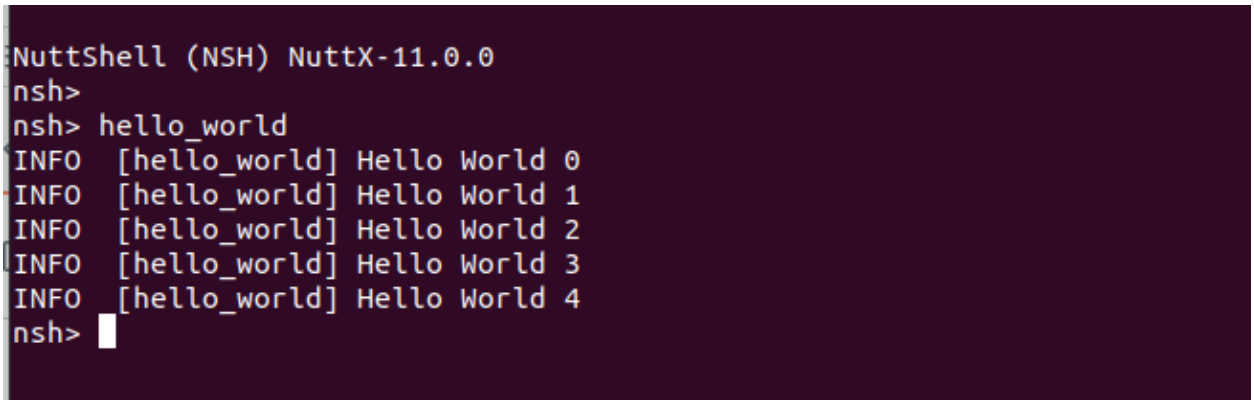
Yuvraj Bal - 400247720

Zhaohan Wang - 400188640

Contributions: All experiments and exercises were performed together and both members contributed equally

Experiment 2

Hello world output every 1 second for five times



Experiment 3A

Output of different messages read

```
 sensor_combined
    timestamp: 525947564 (0.004764 seconds ago)
    gyro_rad: [0.0172, 0.0047, -0.0038]
    gyro_integral_dt: 4992
    accelerometer_timestamp_relative: 0
    accelerometer_m_s2: [0.1010, 0.5065, -9.6100]
    accelerometer_integral_dt: 4986
    accelerometer_clipping: 0

nsh> listener vehicle_acceleration

TOPIC: vehicle_acceleration
 vehicle_acceleration
    timestamp: 616917974 (0.003900 seconds ago)
    timestamp_sample: 616917850 (124 us before timestamp)
    xyz: [0.0490, 0.3607, -9.6980]

nsh> listener vehicle_magnetometer

TOPIC: vehicle_magnetometer
 vehicle_magnetometer
    timestamp: 636714045 (0.056850 seconds ago)
    timestamp_sample: 636687360 (26685 us before timestamp)
    device_id: 4395025 (Type: 0x43, I2C:2 (0x10))
    magnetometer_ga: [-1.2715, -4.6818, -2.6462]
    calibration_count: 0

nsh> listener vehicle_control_mode

TOPIC: vehicle_control_mode
 vehicle_control_mode
    timestamp: 666869743 (0.261458 seconds ago)
    flag_armed: False
    flag_multicopter_position_control_enabled: False
    flag_control_manual_enabled: True
    flag_control_auto_enabled: False
    flag_control_offboard_enabled: False
    flag_control_rates_enabled: True
    flag_control_attitude_enabled: True
    flag_control_acceleration_enabled: False
    flag_control_velocity_enabled: False
    flag_control_position_enabled: False
    flag_control_altitude_enabled: False
    flag_control_climb_rate_enabled: False
    flag_control_termination_enabled: False
```

Experiment 3B

Output of sensor combined message

```
INFO  [hello_world] X = 0.046244, Y = 0.395806, Z = -9.718416
INFO  [hello_world] X = 0.050848, Y = 0.403290, Z = -9.743481
INFO  [hello_world] X = 0.039276, Y = 0.413473, Z = -9.736668
INFO  [hello_world] X = 0.048534, Y = 0.378081, Z = -9.795983
INFO  [hello_world] X = 0.023010, Y = 0.408423, Z = -9.702496
INFO  [hello_world] X = 0.029975, Y = 0.388227, Z = -9.764026
INFO  [hello_world] X = 0.004394, Y = 0.390754, Z = -9.759440
INFO  [hello_world] X = 0.043918, Y = 0.383206, Z = -9.761708
INFO  [hello_world] X = 0.067153, Y = 0.388247, Z = -9.773113
INFO  [hello_world] X = 0.029972, Y = 0.350400, Z = -9.798162
INFO  [hello_world] X = -0.009527, Y = 0.380671, Z = -9.727517
INFO  [hello_world] X = -0.014178, Y = 0.330211, Z = -9.896099
INFO  [hello_world] X = 0.050932, Y = 0.410978, Z = -9.750264
INFO  [hello_world] X = 0.018356, Y = 0.415996, Z = -9.784499
INFO  [hello_world] X = -0.028052, Y = 0.521941, Z = -9.666030
INFO  [hello_world] X = 0.062508, Y = 0.340307, Z = -9.800445
INFO  [hello_world] X = 0.025372, Y = 0.393273, Z = -9.743582
INFO  [hello_world] X = 0.048549, Y = 0.350403, Z = -9.802705
INFO  [hello_world] X = 0.081100, Y = 0.279754, Z = -9.832338
INFO  [hello_world] X = 0.027651, Y = 0.461411, Z = -9.704771
INFO  [hello_world] X = 0.027663, Y = 0.388243, Z = -9.734384
INFO  [hello_world] X = 0.050893, Y = 0.405929, Z = -9.709333
INFO  [hello_world] X = 0.016080, Y = 0.388191, Z = -9.745656
INFO  [hello_world] X = 0.025327, Y = 0.365536, Z = -9.830060
INFO  [hello_world] X = 0.060177, Y = 0.413437, Z = -9.695610
INFO  [hello_world] X = 0.029975, Y = 0.403381, Z = -9.770832
INFO  [hello_world] X = 0.041599, Y = 0.428639, Z = -9.743461
INFO  [hello_world] X = 0.041599, Y = 0.400872, Z = -9.795923
INFO  [hello_world] X = 0.016030, Y = 0.378151, Z = -9.784505
INFO  [hello_world] X = 0.020717, Y = 0.375603, Z = -10.123901
INFO  [hello_world] X = 0.069492, Y = 0.403414, Z = -9.704771
INFO  [hello_world] X = -0.000190, Y = 0.378132, Z = -9.734463
INFO  [hello_world] X = 0.034651, Y = 0.403381, Z = -9.736704
INFO  [hello_world] X = 0.043932, Y = 0.431117, Z = -9.800427
INFO  [hello_world] X = 0.020667, Y = 0.441219, Z = -9.802717
INFO  [hello_world] X = 0.034623, Y = 0.410950, Z = -9.777666
INFO  [hello_world] X = 0.025338, Y = 0.368037, Z = -9.768523
INFO  [hello_world] X = 0.016035, Y = 0.357967, Z = -9.770827
INFO  [hello_world] X = 0.023007, Y = 0.403349, Z = -9.732085
INFO  [hello_world] X = 0.060185, Y = 0.398333, Z = -9.784492
INFO  [hello_world] X = 0.069476, Y = 0.398325, Z = -9.732098
INFO  [hello_world] X = 0.039246, Y = 0.383215, Z = -9.736702
INFO  [hello_world] X = 0.013703, Y = 0.395809, Z = -9.743507
INFO  [hello_world] X = 0.027640, Y = 0.410954, Z = -9.823255
INFO  [hello_world] X = 0.032307, Y = 0.393295, Z = -9.761724
INFO  [hello_world] X = 0.115949, Y = 0.277231, Z = -9.661534
```

Project 2 – step 0

RC value displayed every 200s, range is from –1 to +1

Output:

```
ds@ds-VirtualBox: ~/Documents/PX4-Autopilot
INFO  [rc_receive] channels 5 = 1.000000
INFO  [rc_receive] channels 1 = 0.008163
INFO  [rc_receive] channels 2 = 0.008163
INFO  [rc_receive] channels 3 = 0.071428
INFO  [rc_receive] channels 4 = 0.008163
INFO  [rc_receive] channels 5 = 1.000000
INFO  [rc_receive] channels 1 = 0.008163
INFO  [rc_receive] channels 2 = 0.008163
INFO  [rc_receive] channels 3 = 0.071428
INFO  [rc_receive] channels 4 = 0.008163
INFO  [rc_receive] channels 5 = 1.000000
INFO  [rc_receive] channels 1 = 0.008163
INFO  [rc_receive] channels 2 = 0.008163
INFO  [rc_receive] channels 3 = 0.071428
INFO  [rc_receive] channels 4 = 0.008163
INFO  [rc_receive] channels 5 = 1.000000
INFO  [rc_receive] channels 1 = 0.008163
INFO  [rc_receive] channels 2 = 0.008163
INFO  [rc_receive] channels 3 = 0.071428
INFO  [rc_receive] channels 4 = 0.008163
INFO  [rc_receive] channels 5 = 1.000000
INFO  [rc_receive] channels 1 = 0.008163
INFO  [rc_receive] channels 2 = 0.008163
INFO  [rc_receive] channels 3 = 0.069387
INFO  [rc_receive] channels 4 = 0.008163
INFO  [rc_receive] channels 5 = 1.000000
INFO  [rc_receive] channels 1 = 0.008163
INFO  [rc_receive] channels 2 = 0.008163
INFO  [rc_receive] channels 3 = 0.069387
INFO  [rc_receive] channels 4 = 0.008163
INFO  [rc_receive] channels 5 = 1.000000
INFO  [rc_receive] channels 1 = 0.008163
INFO  [rc_receive] channels 2 = 0.008163
INFO  [rc_receive] channels 3 = 0.071428
INFO  [rc_receive] channels 4 = 0.008163
INFO  [rc_receive] channels 5 = 1.000000
INFO  [rc_receive] channels 1 = 0.008163
INFO  [rc_receive] channels 2 = 0.008163
INFO  [rc_receive] channels 3 = 0.071428
INFO  [rc_receive] channels 4 = 0.008163
INFO  [rc_receive] channels 5 = 1.000000
INFO  [rc_receive] channels 1 = 0.008163
INFO  [rc_receive] channels 2 = 0.008163
INFO  [rc_receive] channels 3 = 0.071428
INFO  [rc_receive] channels 4 = 0.008163
INFO  [rc_receive] channels 5 = 1.000000
```

Part 1 (Revisiting Project 1 using PX4)

(a) Edit the code in Experiment 3C to control both the dc motor and the servo.

Controlling DC and SERVO motor and publishing using the uORB Publication class

```
uORB::Publication<test_motor_s> dc_motor_pub(ORB_ID(test_motor));
uORB::Publication<test_motor_s> servo_motor_pub(ORB_ID(test_motor));

while(1)
{
	PX4_INFO("Enter speed value and servo angle (0 to 1). If you enter a value outside the range,the motor will be stopped and the application will be terminated.");
	scanf("%lf", &motor_value);
	scanf("%lf", &servo_value);
	if(motor_value > 1.0 || motor_value < 0)
		break;
	if(servo_value > 1.0 || servo_value < 0)
		break;
```

```
PX4_INFO("Motor speed is %f, Servo angle is %f", motor_value, servo_value);

test_motor.timestamp = hrt_absolute_time();
test_motor.motor_number = DC_MOTOR;
test_motor.value = (float)motor_value;
test_motor.action = test_motor_s::ACTION_RUN;
test_motor.driver_instance = 0;
test_motor.timeout_ms = 0;

servo_motor.timestamp = hrt_absolute_time();
servo_motor.motor_number = SERVO_MOTOR;
servo_motor.value = (float)servo_value;
servo_motor.action = test_motor_s::ACTION_RUN;
servo_motor.driver_instance = 0;
servo_motor.timeout_ms = 0;

dc_motor_pub.publish(test_motor);
servo_motor_pub.publish(servo_motor);
```

(b) Write a code that integrates reading RC channel, and controlling the motors accordingly

Subscribe to rc_channels to read values from the rc channels and controlling the motors. The remaning code is the same as above experiments

```
int rc_handle;
rc_channels_s rc_data;
rc_handle = orb_subscribe(ORB_ID(rc_channels));
orb_set_interval(rc_handle, 200);

uORB::Publication<test_motor_s> dc_motor_pub(ORB_ID(test_motor));
uORB::Publication<test_motor_s> servo_motor_pub(ORB_ID(test_motor));
```

Part 2

In this part of the project, you will be using the given algorithm above that gives you the distance from the ultrasonic and directions from camera, then write a small logic/function that does the following:

1. Reads the data from sensors, sends them to FMU board and actuates the motors accordingly.
2. It is recommended to have the car decelerates if the distance between the car and the closest obstacle is less than 50 cm, and stops completely if the distance becomes less than 15 cm. (you can define different safe distances to decrease the speed and stop, based on the speed of your car, but please discuss that in the demo and the report

For this part we set the safe distance to be 25cm since dc motor rotates at a high speed The car decelerates between 25 and 70 cm and dc runs with highest speed if nothing is detected in first 70cm. These values can be changed after testing and depending on the application. To read the mavlink data we use Orb subscribe. The mavlink data is sent by the raspberry pi using python code. The code sends two values, distance which is mavlink_data.value and flag for direction using mavlink_data.ind.

The distance is calculated by the ultrasonic function which is calculated by the time difference between the laser leaving the trigger pin and returning back to echo pin. This time is directly proportional to the distance. The time interval is multiplied by the sonic speed and divided by 2 to determine distance in cm. The camera does a few operations on the image like flipping, filtering noise and detecting edges. These edges are then appended to a list for further processing which includes dividing into 3 chunks and determining the average of each chunk. The distance between mid-point of the image frame and mean point of every chunk is used to determine direction as the chunk with shortest distance has no object in the chunk.

```
int debug_handle;
debug_value_s mavlink_data;
debug_handle = orb_subscribe(ORB_ID(debug_value));

orb_set_interval(debug_handle, 200);

uORB::Publication<test_motor_s> test_motor_pub(ORB_ID(test_motor));

while(1){

        if (mavlink_data.value < 25){
                motor_value = 0.5;
        }
        else if (mavlink_data.value < 70 && mavlink_data.value >= 25){
                motor_value = 0.62;
        }
        else{
                motor_value = 0.8;
        }

        if (mavlink_data.ind == 0){ // LEFT
                servo_value = 0;
        }
        else if (mavlink_data.ind == 1){ // FORWARD
                servo_value = 0.5;
        }
        else{ // RIGHT
                servo_value = 1.0;
        }
```